Attorney Docket No. 50325-0525

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:

Maximilian J. Spring

Group Art Unit No.: 2122

Examiner: Kuo-Liang J. Tang

Serial No.: 09/855,386

Filed on: May 14, 2001

For: TECHNIQUES FOR MAINTAINING COMPATIBILITY OF A SOFTWARE CORE MODULE AND AN INTERACTING MODULE

Mail Stop Amendment
Commissioner of Patents
P.O. Box 1450
Alexandria, VA 22313-1450

INTERVIEW AGENDA

(ATTACHMENT TO FORM PTOL-413A)

This is proposed agenda for a personal interview proposed for 14 March 2005.

1. Review of Invention as Shown in Claims (See Claim Amendments)

   a. The references cited are fundamentally different. The claims relate to maintaining compatibility among computer program modules through a software module interface. This is achieved by creating a mapping of the varying interface capabilities at the various release time and version numbers. This mapping of interface capabilities allows interacting modules to determine an applicable version number based on a subset of the stored interface capabilities. The cited art pertains only to updating a router network through BGP messages—a completely different problem—and has no teaching of maintaining compatibility of computer program modules across different platforms.

   b. The Office Action does not fairly consider the specific features of the claims. Applicant understands that the Office takes the broadest reasonable

interpretation of a reference. However, this policy cannot be used to adopt an interpretation so broad that specific claim features are glossed over.

c. The amendment to the claims provides clarification that both the interface and modules described in the claims are computer program modules. Specifically, a core computer program module interface ensures compatibility between interacting computer program modules on different platforms. Examples of the claimed interface and modules are class files expressed in Java.

d. The phrase "computer program module interface capabilities" in Claim 1 is added to clarify that the information that is created and stored describes capabilities and compatibility of the interface core module with interacting modules. Examples of such capabilities include program routines and routine parameters. Interacting modules can determine version capabilities based on matching the subset of the interacting module's capabilities with the interface capabilities stored in the mapping.

2. Review of Cited Prior Art References (See Claim Amendments)

a. Chen Reference—35 U.S.C. § 102(e)

i. Creating and storing information that describe interface capabilities is fundamentally different than the use of the BGP router updating messages in Chen. The BGP messages do not describe computer program interface capabilities at particular release times. Interface capabilities may identify program routines and routine parameters. BGP messages do not contain this information. The description of an interface of a high-level program module is not contained in a network-level BGP message.

ii. Storing the information describing the interface in instances of a data structure is not comparable to FIG. 8, address field 804 contained in the BGP message. The BGP message is discarded upon completion of routing updates.

iii. There is no mapping created between the BGP messages and version numbers. The Office Action asserts that message fields of BGP messages correspond to the instances of a data structure. However,

the BGP messages are not mapped to version numbers and then stored in a computer program module. The claimed mapping pertains to instances of data structures that contain information describing the computer program interface at various release times.

iv. The claims include automatically assigning a second version number to a second module that interacts with the first module by finding at least one instance in the mapping corresponding to the second modules capabilities. Chen FIG. 8 and the associated text do not describe such an automatic assignment of version number to a computer program module. The neighbor version number is used to determine which routers require an update message.

b. RFC 1771. The use of an OPEN message format does not correlate to signature of routines. Signatures of routines as claimed relates to methods or procedures written in a high-level language such as the Java programming language. OPEN message format is a protocol and is fundamentally different.

* * *

AMENDMENTS TO THE CLAIMS

1. (Currently Amended) A method of maintaining version compatibility between a first computer program module and one or more interacting computer program modules that interact with the first module through a interface with capabilities shared by all the interacting modules, wherein the modules are stored in computer storage, the method comprising the computer-implemented steps of:

creating a first information describing the computer program module interface capabilities at one or more plurality of times;

storing the first information in a [[corresponding]] plurality of instances of a data structure wherein each instance of the data structure corresponds to the interface capabilities at one or more plurality of times;

creating and storing a mapping that associates the plurality of instances with a corresponding plurality of version numbers for the first computer program module;

automatically assigning [[developing]] a second version number for a second computer program module of the one or more interacting modules based on a corresponding instance from the plurality of instances of the data structure [[and]] contained in the mapping;

determining compatibility of the modules based on a first version number for the first module and the second version number for the second module.

2. (Currently Amended) A method as recited in Claim 1, wherein the step of automatically assigning [[developing]] comprises:

describing a subset of the interface capabilities, which subset is employed by the second computer program module;

determining from [[of]] the plurality of instances at least one instance including data describing the subset of the interface capabilities; and

assigning the second version number for the second module based on the corresponding version number in the mapping and the at least one instance.

3. (Original) A method as recited in Claim 2, wherein said step of assigning the second version number comprises assigning as the second version number a particular value of the plurality of version numbers for the first module, the particular value associated with a particular instance of the at least one instance, the particular instance corresponding to an earliest time of one or more times corresponding to the at least one instance.

4. (Currently Amended) A method as recited in Claim 1, wherein the plurality of version numbers for the first computer program module corresponding to the plurality of instances vary in one direction with time of the plurality of times corresponding to the plurality of instances.

5. (Currently Amended) A method as recited in Claim 1, wherein said step of describing the computer program module interface capabilities includes generating and storing in a first instance of the data structure data indicating signatures of a plurality of routines of the interface at a first time, wherein a signature of each routine includes a name of the routine and a type of the routine and parameter types for all parameters of the routine.

6. (Original) A method as recited in Claim 1, wherein:

each of the interacting computer program modules include instructions causing one or more processors to obtain at least one of a property of a corresponding networking device type of a plurality of networking devices types and an action performed by the corresponding networking device type;

the first computer program module includes instructions causing one or more processors, based on interacting with a particular interacting computer program module, to perform at least one of communicating with a first device of the corresponding networking device type on a network of networking devices including the first device, and presenting properties of the first device to a network manager for the network, and displaying connections among the networking devices of the network to the network manager; and

the plurality of networking devices types include one or more models of repeater, a switch, a router, a hub, a bridge, and a gateway.

7. (Currently Amended) A method as recited in Claim 1, wherein the second version number is assigned [[developed]] when the second module is developed; and compatibility is determined at a later time.

8. (Currently Amended) A method of maintaining version compatibility between a first module and one or more interacting modules that interact with the first module through an interface, the method comprising the steps of:

retrieving data from a stored mapping between a plurality of instances of a data structure describing the interface at a corresponding plurality of times and a corresponding plurality of version numbers for the first module;

automatically assigning [[developing]] a second version number for a second module of the one or more interacting modules based on the mapping;

determining compatibility based on a first version number for the first module and the second version number for the second module.

9. (Currently Amended) A method as recited in Claim 8, said step of assigning [[developing]] the second version number further comprising the steps of:

describing a subset of the interface, which subset is employed by the second module;

determining of the plurality of instances at least one instance including data describing the subset of the interface; and

assigning the second version number for the second module based on the mapping and the at least one instance.

10. (Currently Amended) A method as recited in Claim 9, wherein said step of assigning the second version number comprises assigning as the second version number a particular value of the plurality of version numbers for the first computer program module, the particular value associated with a particular instance of the at least one instance, the particular instance corresponding to an earliest time of one or more times corresponding to the at least one instance.

11. (Currently Amended) A method as recited in Claim 8, wherein the plurality of version numbers for the first computer program module corresponding to the plurality of instances vary in one direction with time of the plurality of times corresponding to the plurality of instances.

12. (Original) A method as recited in Claim 8, wherein a first instance of the data structure comprises data indicating a routine name and a routine type of a routine of the first module at a first time.

13. (Original) A method as recited in Claim 12, wherein the first data further indicates a parameter type for the routine.

14. (Currently Amended) A method as recited in Claim 8, wherein a first instance of the data structure comprises data indicating signatures of a plurality of routines of the computer program module interface capabilities at a first time, wherein a signature of each routine includes a name of the routine and a type of the routine and parameter types for all parameters of the routine.

15. (Currently Amended) A method as recited in Claim 14, wherein the data indicating signatures of the plurality of routines of the computer program interface capabilities at a first time comprises hashed values, each hashed value uniquely indicating a signature of each routine of the interface.

16. (Original) A method as recited in Claim 14, wherein the plurality of routines comprises all the routines of the interface.

17. (Original) A method as recited in Claim 14, wherein the plurality of routines comprises all the routines of the interface except routines not implemented in the first module.

18. (Currently Amended) A method as recited in Claim 8, wherein:
the second version number is developed when the second computer program module is developed; and

compatibility is determined at a later time.

19. (Currently Amended) A method as recited in Claim 18, wherein compatibility is determined when the second computer program module is installed for use with the first computer program module.

20. (Currently Amended) A method as recited in Claim 18, wherein compatibility is determined when the second computer program module is invoked for execution by the first module.

21. (Currently Amended) A method as recited in Claim 8, wherein each computer program module of the first computer program module and the one or more interacting computer program modules comprises instructions for causing one or more processors to perform one or more tasks.

22. (Currently Amended) A method as recited in Claim 8, wherein the first computer program module comprises instructions for causing one or more processors to manage a plurality of networking devicess in response to data indicating input by a user.

23. (Currently Amended) A method as recited in Claim 8, wherein each interacting computer program module of the one or more interacting computer program modules comprises instructions for causing one or more processors to provide device-specific information for one of a plurality of networking devicess.

24. (Currently Amended) A method as recited in Claim 21, wherein:

each of the interacting computer program modules include instructions causing one or more processors to obtain at least one of a property of a corresponding networking device type of a plurality of networking devices types and an action performed by the corresponding networking device type;

the first module includes instructions causing one or more processors, based on interacting with a particular interacting computer program module, to perform at

least one of communicating with a first device of the corresponding networking device type on a network of networking devices including the first device, and presenting properties of the first device to a network manager for the network, and displaying connections among the networking devices of the network to the network manager; and

the plurality of networking devices types include one or more models of repeater, a switch, a router, a hub, a bridge, and a gateway.

25. (Currently Amended) A method of determining version compatibility between a first computer program module and a computer program second module of one or more interacting modules that interact with the first computer program module through an interface, the method comprising the steps of:

obtaining a first version number for the first module;

obtaining a second version number for the second computer program module, the second version number set when the second computer program module is developed based on a mapping between a plurality of instances of a data structure describing the interface capabilities at a corresponding plurality of times and a corresponding plurality of version numbers for the first computer program module; and

determining whether the computer program modules are compatible based on the first version number and the second version number.

26. (Currently Amended) A method as recited in Claim 25, wherein:

each of the interacting computer program modules include instructions causing one or more processors to obtain at least one of a property of a corresponding networking device type of a plurality of networking devices types and an action performed by the corresponding networking device type;

the first computer program module includes instructions causing one or more processors, based on interacting with a particular interacting computer program module, to perform at least one of communicating with a first device of the corresponding networking device type on a network of networking devices including the first device, and presenting properties of the first device to a network manager for the

network, and displaying connections among the networking devices of the network to the network manager; and

the plurality of networking devices types include one or more models of repeater, a switch, a router, a hub, a bridge, and a gateway.

27. (Currently Amended) A computer-readable medium for maintaining version compatibility between a first computer program module and one or more interacting computer program modules that interact with the first computer program module through an interface, the computer-readable medium carrying:

a plurality of instances of a data structure describing the interface capabilities at a corresponding plurality of times;

a mapping that associates the plurality of instances with a corresponding plurality of version numbers for the first module; and

one or more sequences of instructions, which, when executed by one or more processors, cause the one or more processors to carry out the steps of

retrieving data from the mapping, and

developing a second version number for a second computer program module of the one or more interacting computer program modules,

wherein compatibility is determined based on a first version number for the first computer program module and the second version number for the second computer program module.

28. (Currently Amended) A system for maintaining version compatibility between a first computer program module and one or more interacting computer program modules that interact with the first computer program module through an interface, the system comprising:

means for retrieving a stored mapping between a plurality of instances of a data structure describing the interface capabilities at a corresponding plurality of times and a corresponding plurality of version numbers for the first computer program module;

means for automatically developing a second version number for a second computer program module of the one or more interacting computer program modules based on the mapping; and

means for determining compatibility based on a first version number for the first computer program module and the second version number for the computer program second module.

29. (Currently Amended) A computer system for maintaining version compatibility between a first computer program module and one or more interacting computer program modules that interact with the first computer program module through an interface, the system comprising:

a processor;

a computer-readable medium carrying

a stored mapping between a plurality of instances of a data structure describing the interface capabilities at a corresponding plurality of times and a corresponding plurality of version numbers for the first computer program module, and

one or more stored sequences of instructions which, when executed by the processor, cause the processor to carry out the steps of:

retrieving data from the mapping;

developing a second version number for a second computer program module of the one or more interacting computer program modules based on the mapping; and

determining compatibility based on a first version number for the first computer program module and the second version number for the second computer program module.